Working with APIs

Overview

An API defines the REST or web interface to services provided by a service owner. In order to use a REST service, the owner must provide you with some type of documentation for the service interface, either through an OpenAPI (commonly known as 'Swagger') API definition, or written descriptions of the paths, methods, parameters and body content required to use the service.

> In this document, we use the term "REST service" to refer to web services offering REST or web APIs over HTTP, whether or not they fully conform to REST (REpresentational State Transfer; Roy Fielding) conventions. You may find owners of these services to use different terminology, sometimes inaccurately, to refer to the same thing: "REST service", "REST API", "web API", etc.

LightWave Client requires an API definition in order to facilitate access from client applications on NonStop servers to remote REST services. You use the LightWave Client Console browser application to create, view and edit API definitions. You can create an API definition by importing a Swagger definition, or using the Console's API Editor together with example requests and responses, either of which may be provided by the service owner. You may also import an API definition previously created and exported from LightWave Client.

API Definitions

An API *definition* consists of

- an API name
- a description
- a list of operations

An *operation*, in turn, is comprised of

- a URI path (e.g. /employees/{employee-id}
- one or more methods

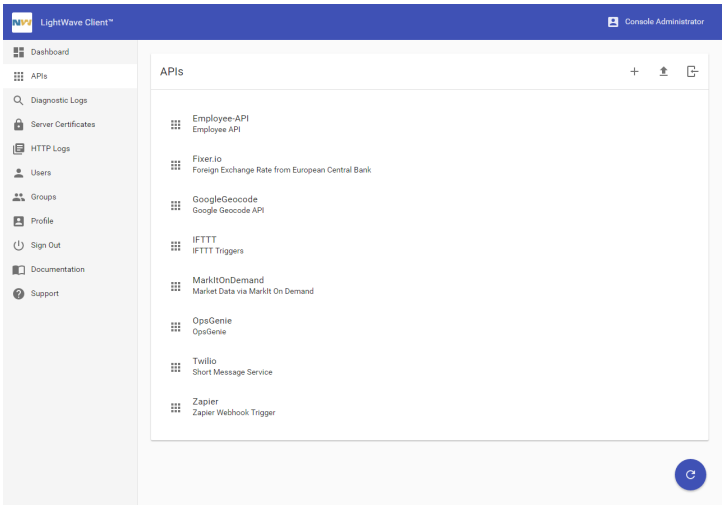
A *method* is comprised of

- an HTTP verb (GET, POST, etc.)
- a textual description (optional)
- I/O options
- a request definition
- one or more response definitions

Each request and response definition may include zero or more mappings. The request mappings determine how LightWave Client transforms data received in the client application request IPM into the HTTP request that will be sent to the remote service endpoint. Likewise, response mappings determine how LightWave Client transforms the data in the HTTP response from the service endpoint into the IPM that will be replied to the client application. Since the REST/web API may return differently formatted responses depending on the outcome of the request, each method may include more than one response definition.

Using the Console

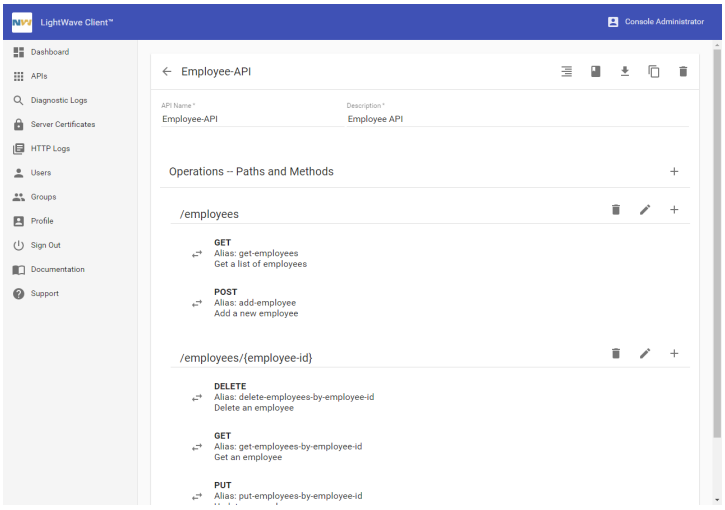
To begin using the LightWave Client Console to work with APIs, sign in to the Console, then use the menu to navigate to APIs. The list of existing APIs is displayed.



Use the view menu in the upper right corner to:

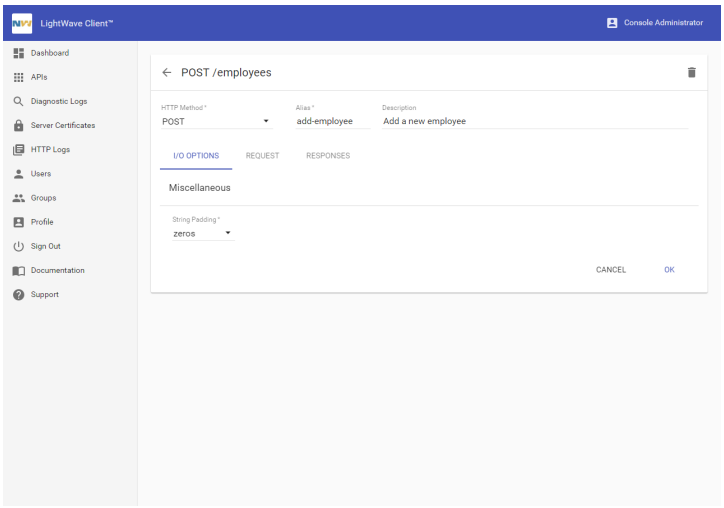
- Create an API using the Editor
- Import an API previously created and exported by LightWave Client
- Create an API by importing a Swagger definition

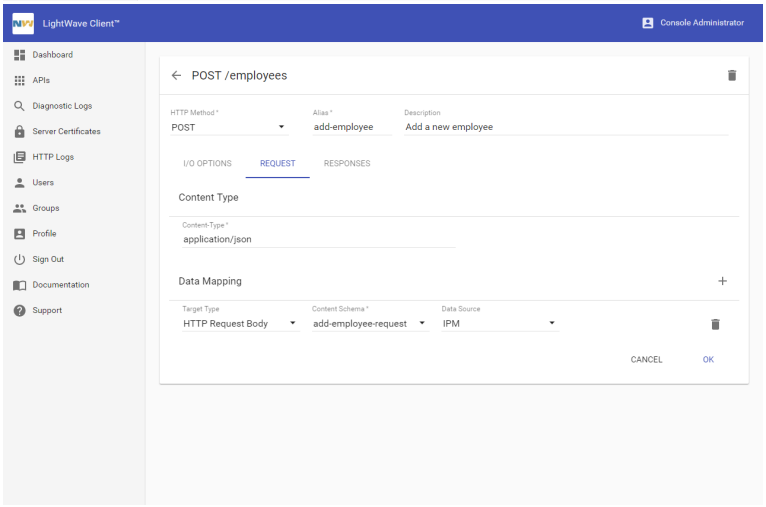
To view the definition of an API, click the name of the API to open the API Editor for the selected API.

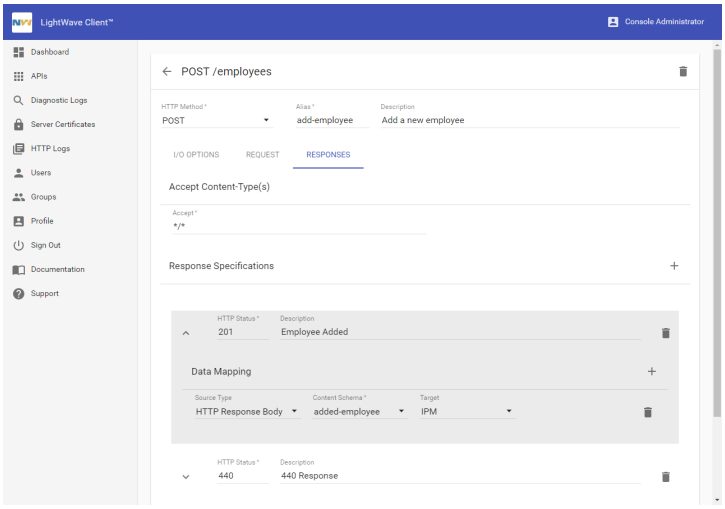


Using the API Editor

The API Editor is used to create, view and modify API definitions. Each API consists of one or more operations. An operation is the combination of a path and a method (an HTTP verb, e.g. GET or POST). An operation defines an HTTP request and response.







Each operation is defined to have a single request format, for which multiple response formats can be defined. Each request and response can include zero or more data item mappings.

I/O Options

StringPadding specifies how LightWave will treat character string fields in interprocess messages (IPMs).

setting	description

zeroes	A setting of zeros means that LightWave will expect the client application to zero-terminate (or "null-terminate") strings in IPMs it sends to LightWave, and LightWave will zero-terminate strings in IPMs it returns to the client application. This setting is typical for client applications written in the C/C++ programming language.
spaces	A setting of spaces means that LightWave will expect the client application to blank pad strings in IPMs it sends to LightWave, and LightWave will blank pad strings in IPMs it returns to the client application. This setting is typical for client applications written in the COBOL programming language.

Request

ContentType

ContenType specifies the value of the Content-Type HTTP header that will be sent with the request to the web service. The value should be a media type name (see Media Types). If the request specifies an HTTP Request Body mapping (see below) which is set to a schema type other than "BLOB", the media type should be "application/json" to indicate a JSON payload or "text/xml" to indicate an XML payload . If the schema type is "BLOB", set the value to a media type that describes the content of the BLOB, which should be a type that the web service accepts. If the request specifies one or more HTTP Form Field mappings, the value should be "application/x-www-form-urlencoded".

Data Mapping

LightWave performs mapping of data items from interprocess messages (IPMs) that client applications send to various components in the HTTP request that is sent to the web service. Depending on the component of the HTTP Request being mapped to, there are additional parameters required.

target	description	name	source
URI Path Component	a component of the path for the current method; where the path has been defined to contain one or more 'path parameters', e.g. /employees/{employee-id}, where {employee-id} is a path parameter. At runtime, LightWave will construct the request path by replacing any path parameter placeholders with the corresponding mapping's 'source' value.	the path parameter name	IPM, constant, API parameter
URI Query Parameter	a name=value pair appended to the method's request path, e.g., /employees?format=detailed&sort=ascending, where 'format' and 'sort' are query parameter names.	the query parameter name	IPM, constant, API parameter
HTTP Form Field	a name=value pair that is encoded and combined with other form fields and placed in the request message body "content". This mapping type is mutually exclusive with "HTTP Request Body". See also ContentType, above.	the form field name	IPM, constant, API parameter
HTTP Request Header	a name:value pair that is added to the list of headers in the request message to be sent to the web service	the header name	IPM, constant, API parameter
HTTP Request Body	an object in Javascript Object Notation described by a schema. A schema can be added 'manually', or 'by example' using sample request content (typically provided by the web service owner). A maximum of one HTTP Request Body mapping is allowed, and is mutually exclusive with "HTTP Form Field" mappings. Alternatively, the request body content schema can be defined as "BLOB", in which case LightWave inserts the client-application supplied data into the request body verbatim. See also ContentType, above.	a schema (type) name or BLOB	IPM, FILE

source	description
IPM	the source value is taken from a field in the request IPM. The maximum length of the value must also be specified.
constant	the source value is taken from the constant value specified in the mapping
API parameter	the source value is taken from a command line parameter supplied to the LightWave CLIENT process. See Working with API Parameters.
FILE	the source content is taken from a file, the name of which is supplied by the client application in the request IPM

Responses

You may define one or more responses for each request, each response corresponding to an HTTP status code that may be returned from the called web service.

Accept ContentType

Specifies the value of the Accept HTTP header that will be sent with the request to the web service. The value should be a media type name (see Media Types), or a comma-separated list of types. Wildcards are allowed. The default value is ***/***, which is suitable in most cases. However, some web services provide responses in different formats according to the Accept header (for example, JSON vs. XML). In such cases, it may be necessary to specify a value other than the default ("application/json", for JSON format, for example). See W3C HTTP Header Field Definitions, section 14.1 Accept for more information.

Data Mapping

LightWave performs mapping of data items from the called web service's HTTP response into fields in the interprocess messages (IPM) that will be returned to the client application. Zero or more mappings may be defined for each response.

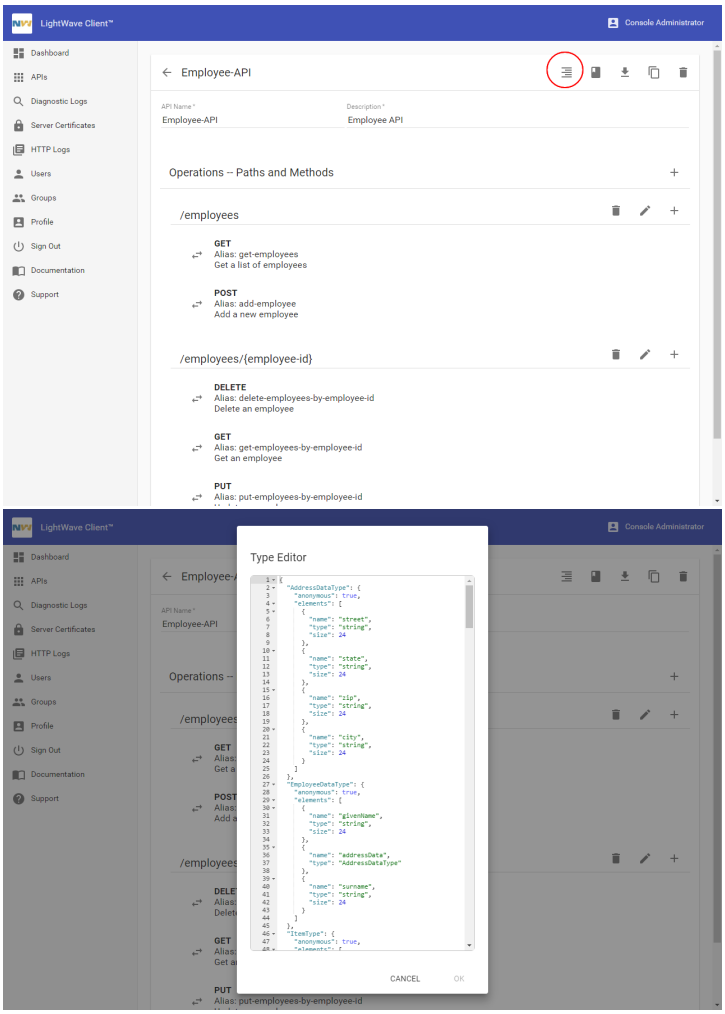
source	description	name	target
HTTP Response Header	a name:value pair in the list of headers contained in the response from the web service. The maximum size of the value must also be specified.	the header name	IPM
HTTP Response Body	an object in Javascript Object Notation described by a schema. A schema can be added 'manually', or 'by example' using sample response content (typically provided by the web service owner). A maximum of one HTTP Request Body mapping is allowed. Alternatively, the request body content schema can be defined as "BLOB", in which case LightWave stores the raw body content in an Enscribe file and copies the name of the file to a field in the IPM that is returned (REPLY-ed) to the client application.	a schema (type) name or BLOB	IPM, FILE

target	description
IPM	the response value is stored in the IPM
FILE	the response body content is stored in a file, the name of which is returned in a field in the IPM. See --blob-files command line parameter for the LightWave CLIENT process for related information.

Importing a Swagger (OpenAPI) Definition

You can create an API by importing a Swagger (aka OpenAPI) definition, either by entering the location of the definition by URL, or by pasting the definition into the text box. LightWave Client translates the Swagger definition into LightWave Client format. You may choose to edit the imported definition to adjust to your liking the names of certain items (e.g., method aliases or type names) that were generated by LightWave

Annotating Data Types



The request and response message content (or 'body') of REST/web APIs is composed of Javascript primitive data types (string, number, boolean, null), objects and arrays (which themselves can contain primitives). Certain primitive types do not translate cleanly to NonStop data types. For example, Javascript strings are inherently of unlimited length, whereas NonStop strings must be allocated to have a finite maximum length. To accommodate this, LightWave allows data type descriptions to be 'annotated' with additional properties:

```
"elements": [
  {
  "name": "state",
  "type": "string",
  "size": 2
  },
  {
  "name": "zipcode",
  "type": "string",
  "size": 9
  }
]
```

See Working with Schema for information on the available element properties.